ANKARA ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM 4061 GÜZ DÖNEMİ PROJE RAPORU

3D Reconstruction Using Drones

Ömer Mert Erdal
18290739

Alperen Yılmaz
18290071

05.07.2021 – 02.01.2022

Danışman: Gazi Erkan Bostancı

ÖZET

Bu rapor Ankara Üniversitesi Bilgisayar Mühendisliği öğrencileri Ömer Mert Erdal ve Alperen Yılmaz tarafından bitirme projesi için güz dönemi final raporu olarak hazırlanmıştır. Rapor “3D Reconstruction Using Drones” yani drone kullanarak 3 boyutlu yeniden yapılandırma konusu anlatılmakta ve nasıl gerçekleştirileceğine yönelik yol gösterilmektedir. Proje sırasında simülasyon ortamı olarak Airsim kullanılmıştır. OpenCV kullanılarak simülasyon ortamından görüntüler elde edilmiş ve Meshroom yazılımı bu görseller üzerinde çalıştırılarak bilgisayar ortamında modelleme işlemi yapılmıştır.

İÇİNDEKİLER

1. Giriş

Nesnelerin 3D rekonstrüksiyonu, Bilgisayar Destekli Geometrik Tasarım (CAGD), bilgisayar grafikleri , bilgisayar animasyonu , bilgisayarla görme , tıbbi görüntüleme , hesaplamalı bilim , sanal gerçeklik , dijital medya gibi çok çeşitli alanların genel olarak bilimsel bir problemi ve temel teknolojisidir. [1]

3 Boyutlu Yeniden Yapılandırma temel olarak nesnelerin görünümünü yakalama işlemidir. 3B verileri öğrenmek, saklamak ve görsel olarak daha iyi hale getirebilmek için bu bilgilerden faydalanırız. Bu işlem pek çok yöntemle yapılabilir. 3 Boyutlu Yeniden Yapılandırma alanında kabul gören 2 ana yaklaşım bulunmaktadır. Bunlar derinlik tabanlı yöntemler kullanan Lidar sistemleri veya fotogrametri ve bilgisayar görüsü yöntemleri kullanılan görüntü bazlı yeniden yapılandırma yöntemleridir. [2] İki yöntemde de yapılan işlemin amacı hedeften yararlanarak bilgisayar ortamına yeterli sayıda nokta kümesi toplamak ve bu toplanan noktalardan faydalanarak hedefi bilgisayar ortamında yeniden oluşturmak olmakla beraber çalışma prensibi açısından temel farklılıklara sahiplerdir.

Raporun Literatür Taraması başlığı altında projenin hazırlık aşamasında yapılan literatür taraması işlemlerinden kısa bir kesit sunulmuştur. Materyal-Yöntem Seçimi başlığı projenin hayata geçirilirken kullanılabilecek yöntemlerin tanıtılması ve hangi yöntemin niye seçildiğini anlatır. Yapılan Çalışmalar başlığında projenin yapılış süreci ve bu süreçte elde edilen çıktıları içerir. Sonuç başlığı ise projeden elde edilen sonucu ve bu sonucun değerlendirilmesini içermektedir.

2. Literatür Taraması

2.1. ARES: Accurate, Autonomous, Near Real-time 3D Reconstruction using Drones

Bu çalışmada amaç büyük 3 boyutlu yapıların insan müdahalesi olmadan gerçek zamanlıya yakın bir sürede modellenmesini sağlamaktır. Modelleme çalışmalarında Lidar kullanılan sistemlerde toplanan verilerin birleştirilebilmesi için referans noktalar alınması gerekmektedir ve bu referans noktaların kalitesi modelin sonuç kalitesini önemli şekilde etkilemektedir. Bu problemin çözümü için SLAM algoritmaları alternatif olmaktadır. Bu algoritmalar art arda toplanan Lidar verileri için eşleştirmeler sağlayarak tüm verilerin başlangıç koordinatlarına göre hizalanması sağlanmaktadır.

Ama SLAM algoritmaları uzun uçuşlarda veya uçuş planının optimal olmadığı durumlarda iyi çalışmamaktadır. Bu problemlerin çözümü için ARES sistemi önerilmiştir. ARES sistemi öncelikle yüksek irtifadan hızlı bir uçuş yaparak hedefin sınırlarını tespit eder. Bu sayede rota hesaplanırken hedefin dışındaki ögelere minimum zaman harcanır. Daha sonra bir uçuş planı oluşturulur ve SLAM algoritmalarıyla veri toplanır fakat buradaki farklılık uçuşlar sırasında SLAM algoritmalarında oluşan kaymaların uçuş sırasında tespit edilerek rotanın yeniden ayarlanmasını sağlar. Son olarak bu uçuş sırasında toplanan verilerin işlenmesi iha üzerinden alınarak kablosuz bağlantı üzerinden uzaktaki sunuculara yönlendirilir bu sayede hesaplama donanımının iha üzerinde yük oluşturmasının önüne geçilir ayrıca sunucularda iha üzerinde taşınacak donanımdan çok daha güçlü donanımlarla işlemler yapılabildiği için verilerin modele dönüştürülmesi gerçek zamana yakın hızlarda başarabilmiştir.[3]

2.2. Annotated Reconstruction of 3D Spaces Using Drones

Bu çalışmada ana odak 3 boyutlu uzayların 2 boyutlu görsellerden modellenen ortamdaki nesnelerin etiketlenmesi problemine Faster RCNN nöral ağ yönteminin kabul edilebilir bir doğrulukla iha donanımında çalışabilecek kadar optimize bir versiyonu ile çözüm oluşturmaktır. Nöral ağ sayesinde ortamdaki nesneler ayırt edilebilmektedir. Nesneleri ayırt edilmiş ortam üzerinde 3d modelleme

algoritması çalıştırılmıştır. Bu sayede modellenen ortamdaki nesneler etiketlenmiştir.[4]

2.3. Fotogrametrik Harita ve LiDAR Verileri ile 3B Kent Modeli Üretimi

3 boyutlu kent modellerinin standartlaşması için CityGML standartları geliştirilmiş. CityGML standartlarına göre, modellemeler özelliklerine göre beş ayrıntı düzeyinde (LoD) tanımlanmış. Bu çalışma, LoD2 yani çatı yapıları dahil bina modelini baz alarak hazırlanmış. Bu standartlar sayesinde modellerin farklı platformlara aktarılabilmesine olanak sağlanmış. Google Earth, BingMaps vb. servisler buna örnek olabilir fakat bu servisler LoD1, LoD2 gibi kaba bir ayrıntı düzeyinde kalmış.

Bu çalışmada, İstanbul'un Beyoğlu ilçesi Tophane ve Karaköy semtleri içerisinde bir alanda ArcGIS ve CityEngine yazılımları kullanılarak 3B kent modeli elde edilmiş. Bu model fotogrametrik ve hava Lidar verileri yardımıyla Sayısal Yükseklik Modeli ve Sayısal Yüzey Modeli oluşturulup bunlardan yararlanılarak üretilmiş.

Modeli üretimi için prosedürel modelleme yaklaşımı izlenmiş. Prosedürel modellemede binalar için önce dış hatlar yani dış cephe yapılandırmaya başlar daha sonra pencereler, kapılar için ayrıntılar eklenir.

3B modelleri oluşturmak ve yönetmek için CityEngine, ESRI, Infrastructure Design Suite, Autodesk, Bentley Map, Bentley ve GeoMedia 3D Hexagon Geospatial gibi farklı araçlar kullanılmaktadır. Bu çalışmada model üretimi için CityEngine yazılımı kullanılmış. CityEngine, coğrafi verileri kullanarak bütün bir şehri modelleyebilen ve kullanıcı dostu bir sistemdir. Kentsel ortam sıfırdan oluşturulur. Modeli oluşturan bitki örtüsü, sokak görünümü ile kent alanına bilgi eklemek SYM, 3B model vb. ile başlanır. En son estetik unsurlar eklenir.

[3B model sayısal fotogrametrik verilerinden elde edilen binalar, ağaçlar, deniz yüzeyi; Lidar verilerinden üretilen SYM, YM, ortofoto; ArcGIS'in sağladığı topografik harita ve Open Street Map (OSM) veri sağlayıcısından elde edilen yol verileri kullanılarak üretilmiştir. İlk olarak çalışma alanında bulunan objeleri içerecek şekilde ArcGIS ortamında oluşturulan veri tabanı CityEngine yazılımına aktarılmış ve

CityEngine’in CGA (Computer Generated Architecture) tabanlı kural dosyalarından (rule file) yararlanarak bina ve ağaçların 3 boyutlu modelleri elde edilmiştir. Bu modellere deniz yüzeyi ve OSM’den indirilen yol verileri eklenerek 3B kent modeli üretimi tamamlanmıştır. Binalara çatı ve yüz modelleme uygulamaları gerçekleştirilerek kent modeli estetik olarak gerçekçi bir görünüme kazandırılmıştır. Ve son olarak çalışmanın son kullanıcılarla paylaşılması için City Engine – Web Scene, Google Earth ve Web Viewer gibi ortamlara farklı LoD düzeylerinde aktarılmıştır.] [5]

2.4. LiDAR verilerinin CSF algoritmasıyla filtrelenmesi ve Sayısal Arazi Modeli üretimi

Lidar verisi bir nokta kümesidir. Bu kümenin içerisinde yer yüzeyini ve objeleri temsil eden noktalar bulunur. Bu kümedeki noktalardan yer yüzeyini temsil eden noktaların, diğer cisimleri temsil eden noktalardan ayrılması işlemine filtreleme denir.

Bu çalışmada da Lidar ile elde edilmiş nokta kümesinden CSF algoritması kullanılarak yer yüzeyi filtrelenmiş bu sayede Sayısal Arazi Modeli olarak adlandırılan veri üretilmiştir. Üretilen bu sayısal arazi modeli 3B haritalandırmada kullanılan temel bir bileşendir. Arazi modeli nokta tabanlı ve görüntü tabanlı karşılaştırılmıştır. Karşılaştırılma sonucunda nokta tabanlı filtreleme işlemi %85 üretici doğruluğuna sahip olduğu belirtilmiş. Görüntü tabanlıda karşılaştırma sonucu ise kullanılan arazi modeli ve üretilen model arasındaki korelasyon yaklaşık %98 olarak hesaplandığı belirtilmiş. Ek olarak yükseklik bilgisinin güvenilirliği için hesaplanan karesel ortalama hata 11 cm olarak bulunduğu söylenmiş. Daha sonra da hesaplamalarda kullanılan mantık basit olarak açıklanmış. İstanbul büyükşehir belediyesinden alınan veri seti kullanılmış. Farklı tipte ağaç ve yerleşim yeri bulun bir alanda yapılmış. Sonuç olarak CSF algoritması %85 doğruluğa ulaşırken kalite değeri olarak %62 de kalmıştır. Bu durum sebep olarak yer yüzeyi olmayan düşük yükseklikli objeleri de yer noktası olarak kabul etmesinden kaynaklandığı belirtilmiş.[6]

2.5. İHA ve yersel fotogrametrik veri füzyonu ile kültürel mirasın 3 boyutlu (3B) modelleme uygulaması: Kanlıdivane Örneği

Bu çalışmada Fotogrametri tekniği kullanılmış. Hem yerden elde edilen resimler ile fotogrametri hem de İHA fotogrametrisi tekniği kullanılarak çalışmalar yapılmış. İki teknik kullanılarak veriler birleştirilmiş ve Kanlıdivane sarayı foto gerçekçi doku kaplanarak 3B yüzey modellemesi gerçekleştirilmiş.

Ölçülen kontrol noktalarının hesaplanması ve hazır hale getirilmesi için Netcad yazılımı kullanılmış, fotogrametrik değerlendirme için Agisoft Photoscan yazılımı kullanılmış, yerden ve havadan çekilen fotoğraflardan üretilen verilerin birleştirilmesi için Cloud Compare yazılımı kullanılmış. Fotogrametrik blok dengelemesi için Hareket Tabanlı Yapısal Algılama (Structure from Motion) tekniği kullanılmış. SfM tekniği ile oluşturulan modellerin gerçeğe yakın ölçüm için uygun olduğu bulunmuş.

Elde edilen modellerin karesel ortalama hataları 1.306 cm, yersel fotogrametri veri füzyonu 1.330 cm, İHA fotogrametrisi veri füzyonu 1.373 cm olarak hesaplanmış. [7]

2.6. İHA ve Yersel Fotogrametrik Teknikler Kullanarak Aksaray Kızıl Kilise'nin 3 Boyutlu Nokta Bulutu ve Modelinin Üretilmesi

Aksaray ili içinde bulunan Kızıl Kilise'nin 3B modellenmesi İHA görüntüleri ve fotogrametri tekniği kullanılarak yapılmış. Proje şu 3 aşamadan oluşuyor; verilerin Photomodeler UAS yazılımında değerlendirilmesi, verilerin Agisoft Photoscan yazılımında değerlendirilmesi ve doğruluk analizi.

İlk olarak Photomodeler yazılımına çekilen fotoğraflar ve detay noktası olarak adlandırılan veriler tanıtılmış. Bu detay noktaları fotoğraflar üzerinde manuel olarak işaretlenmiş ve birleştirilmiş. Ardından dengelenme işlemi yapılmış ve 3B iskelet elde edilmiş. Sonra bu modele çekilen fotoğraflardaki yüzeyler giydirilmiş ve 3B model oluşmuş.

Agisoft PhotoScan yazılımı Hareket ile Nesne Oluşturma (Structure from Motion-SFM) tekniğini kullanmaktadır. SFM tekniği nesnenin farklı açılardan çekilmiş çok sayıda fotoğrafını alıp nesneyi sanal ortamda 3B olarak modelleme

yöntemidir. SFM yöntemi ile model geometrisi, kamera pozisyonu ve oryantasyon bilgisi aynı anda otomatik olarak hesaplanır.

Ortalama konum hatasını; Photomodeler yazılımında ±20.4 mm ve Agisoft yazılımında ±17.1 mm bulunmuş.[8]

2.7. İnsansız Hava Aracı Kullanılarak Antık Kent Ve Tarıhı Kervan Yolunun Fotogrametrık Belgelenmesi: Sarıhacılar Örneği

Bu çalışma Antalya İli, Akseki ilçesi Sarıhacılar Köyü arasındaki kervan yolu ve Sarıhacılar kentsel sit alanında yapılmış. İHA ile elde edilen görüntüler SMF algoritması ile işlenmiş, ortofoto görüntüsü ve sayısal yükseklik modeli üretilmiş. Sistemin doğruluğu geleneksel yöntemlerle ölçülerek test edilmiş. İHA fotogrametri yönteminin yeterli hassaslık ve detaylı 3B modelleme için uygun olduğu sonucuna varılmış.[9]

3. Materyal-Yöntem Seçimi

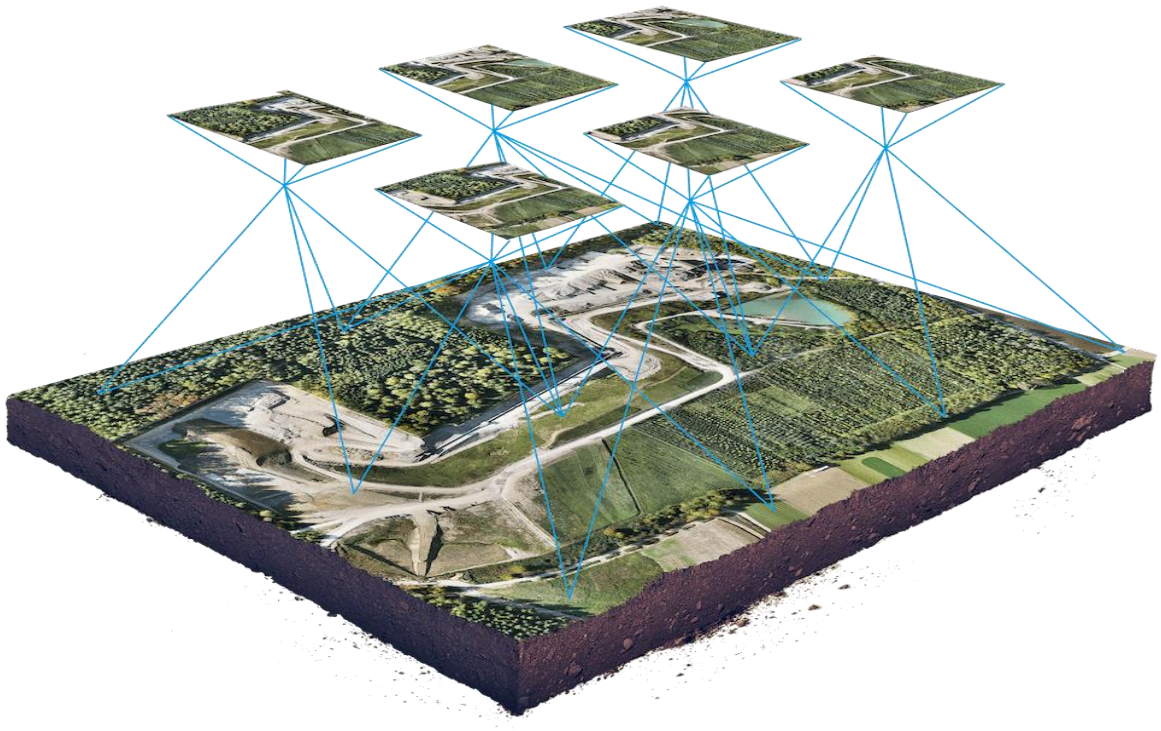
3.1. Fotogrametri ve LİDAR Karşılaştırması

Fotogrametri ve Lidar'ın üç boyutlu çıktılar üzerindeki farklarını 4 başlık altında inceleyeceğiz:

1. Fotogrametri nedir? Nasıl çalışır?
2. LİDAR nedir? Nasıl çalışır?
3. Doğruluk kıyaslaması
4. Fotogrametri ve LIDAR maliyet karşılaştırması

3.1.1. Fotogrametri Nedir? Nasıl Çalışır?

Fotogrametri, tanımsal olarak belirli bir alan veya bir cisim üzerindeki yüksek çözünürlüklü fotografik görüntüleri kullanarak bu nesnenin ölçme ve yorumlama sürecine yardımcı olacak bilgi edinme teknolojisidir.



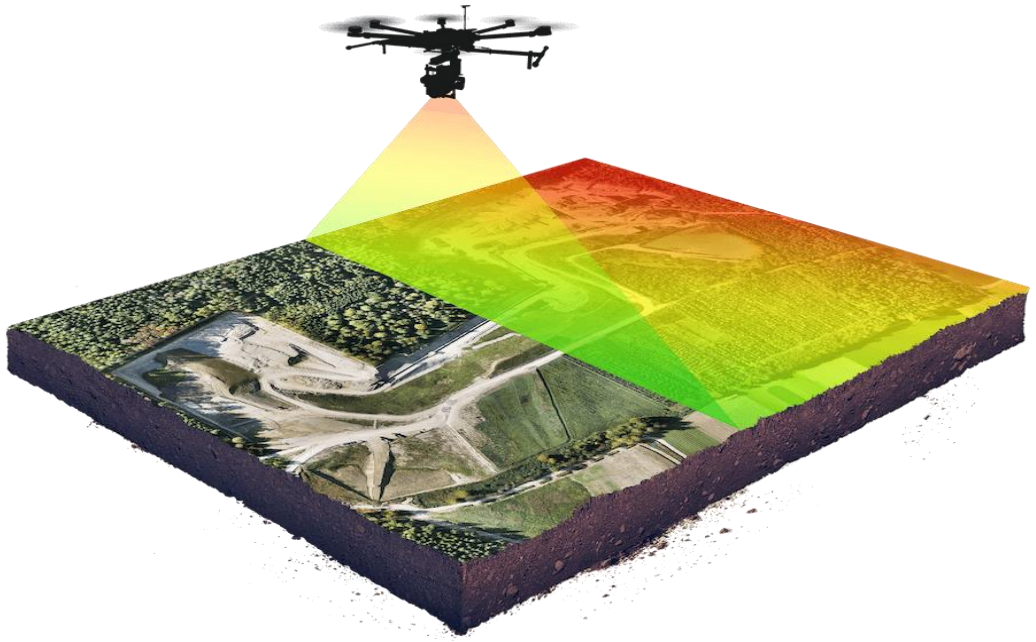
Şekil 1. Fotogrametri Temsili Çalışma Görseli

Bu proje için ise bir drone aracılığıyla, bir alan üzerinde çok sayıda fotoğraf ile alanı farklı bakış noktalarından görüntülememizi sağlayacak şekilde fotoğraflar elde

etmemiz gerekmektedir. İnsan beynindeki derinlik algısını sağlamak için gözlerin gelen bilgiyi kullanmasına benzer şekilde, fotogrametri de 3b bir harita oluşturmak için görüntülerdeki bu çoklu bakış açılarını kullanır. Bu sayede haritadaki her bir nokta ölçülerek çeşitli özelliklerin elde edilmesine olanak sağlar. Elde edilen 3B nokta bulutunun daha kolay yorumlanabilmesini sağlar.[10]

3.1.2. LİDAR Nedir? Nasıl Çalışır?

LİDAR (light detection and ranging), lazer darbeleri kullanılarak bir nesne veya bir yüzeyin uzaklığını ölçmeye yarayan teknolojidir. Bir Lidar sensörü, lazer ışığı atımları gönderir ve bu atımların yüzeyden sekerek geri dönmeleri için geçen süreyi ölçer. Bu ışınlardan yansıyanlar kullanılarak cisme ait mesafe bilgisinin elde edilmesi prensibiyle çalışır. Aynı zamanda bu yansımanın yoğunluğunu da ölçer. Lazer ışınları kullandığı için sürekli hareket halinde çalışmaya ihtiyaç duyar. Etrafını sürekli tarar ve çok sayıda lazer ışını göndererek, milyonlarca noktadan oluşan bir nokta bulutu meydana getirir. Lidar kullanan yeniden yapılandırma yöntemlerinde veri olarak tarayıcının konumuna göre belirli uzaklıklarda noktalar oluşur ve bu noktaların kümelenmesi sonucu hedefin yeniden yapılandırılması gerçekleştirilir.[11]



Şekil 2. Lidar Taraması İçin Temsili Çalışma Görsel

3.1.3. Doğruluk kıyaslaması

Gördüğümüz gibi, fotogrametri ve hava Lidar , yerdeki noktaların kaydedilme biçiminde farklılık gösterir. Bu, nihai nokta bulutu doğruluğunu doğrudan etkiler ve özellikle yoğun orman gölgeliklerinden arındırılmış alanların yatay doğruluğu için fotogrametrinin hava Lidar 'ından açıkça daha iyi performans gösterdiğini göreceğiz. [10]

Hassasiyet açısından Lidar bazlı sistemler fotogrametri bazlı sistemlere göre daha üstün performans göstermekle beraber Lidar sistemleri özel kurulumlar gerektirmekteyken günümüzde neredeyse bütün telefonlarda kamera bulunması görüntü bazlı yeniden yapılandırma yöntemlerini çok daha erişilebilir kılmaktadır. [12] Fakat geniş alanların yeniden yapılandırılması için hava araçları kullanılan senaryolarda kamera tabanlı sistemler Lidar kullanan sistemlere göre performans açısından yetersiz kalmaktadır.

3.1.4. Maliyet karşılaştırması

Çoğu durumda, tek başına bir Lidar sensörü, tüm fotogrametri veri toplama sisteminin maliyetine rakip olacaktır. Hassas coğrafi konumlu lazerler, kameralardan daha pahalıdır. Bu nedenle, LIDAR'a yapılan yatırımın en iyi karar olduğundan emin olmak için mevcut ve gelecekteki uygulamalarınızı değerlendirmek çok önemlidir.[10]

Fotogrametri kullanan drone sistemleri uygun maliyetlidir. Verileri nerede ve ne zaman elde edeceğiniz konusunda esneklik sağlar. Bu yüzden daha kullanışlıdır diyebiliriz.

3.3. Simülasyon Ortamları

3.3.1. ROS Nedir ?

Ros açılımı Robot Operating System olan ve robotların kontrol edilmesini sağlayan bir yazılımdır. Sensörlerden alınan verilerin işlenmesi veya robotun komutları yerine getirmesi gibi işlemlerin yapılabilmesine olanak sağlar. ROS düğüm denilen yapılar içerir ve yapılmak istenen işlemler birbirinden bağımsız düğümler halinde sisteme entegre edilir. Düğüm yapısının diğer bir avantajı da farklı

programlama dilleriyle yazılmış düğümlerin birbiriyle entegre çalışmasına izin vermesidir.

Ros publish-subscribe mesajlaşma modeli kullanarak düğümler arasında iletişim kurar.[13]

3.3.2. Gazebo Nedir ?

Bir veya birden fazla robotun 3 boyutlu ortamda simüle edilebilmesini sağlayan açık kaynaklı bir simülasyon ortamıdır. Kullanıcıların kendi robotlarını tasarlamasına izin verdiği gibi önceden hazırlanmış robotların kullanılmasına da izin verir. Ayrıca kullanılan robotlar sensörler içerebilir ve bu sensörlerin gerçek hayata yakın şekilde simüle edilmesi sağlanabilir. Gazebo'nun en büyük avantajlarından birisi ROS ile entegre çalışarak robotun üzerinde çalışacak yazılımla birlikte simüle edilmesine izin vermesidir. Bu sayede simülasyon ortamında gereklilikleri sağlayan robotların gerçek hayata geçirilmesi kolaylaşmaktadır.[14]

3.3.3. Airsim Nedir ?

Airsim dronlar ve araçlar gibi şeyleri simüle etmek için yapılmış bir simülasyon ortamıdır. Microsoft tarafından açık kaynaklı olarak geliştirilmiştir ve Gazebonun aksine Windows ortamında çalışabilmektedir. Gazebodan ayrıldığı bir diğer temel fark ise Unreal Engine adlı oyun motoru üzerine kurulmuştur. Ayrıca Gazebo gibi Airsim de ROS ile entegre çalışabilmektedir.[15] Airsim simülasyon ortamının projede sağladığı en büyük avantaj kurulum işleminin basit olması ve Python API ye sahip olması sayesinde hızlıca simülasyonu kullanarak geliştirme yapılmasına imkan sağlamasıdır. Airsim'de kullanılan araçların özellikleri bir json dosyasını düzenleyerek belirlenebilir. Bu sayede araç üzerine kolaylıkla kameralar, sensörler eklenebilir. Ve bu donanımların ayarları gene aynı dosya içinden yapılabilir.

3.3.4. OpenCV Nedir ?

OpenCV açık kaynaklı bir bilgisayarlı görü kütüphanesidir. 2000 yılından beri BSD açık kaynak lisansı altında yayınlanmıştır ve bilgisayarlı görü problemlerini çözmek için gerekli araçları sağlamayı amaçlar. Temel görüntü işleme

fonksiyonlarından yüz tanıma, özellik eşleştirme ve görüntü takibine kadar bir çok alanda algoritmalar ve fonksiyonlar içerir. [16]

Büyük çoğunlukla C++ programlama dili ile birlikte kullanılsa da Python, C gibi diğer diller için de destek vermektedir. Asıl odağı gerçek zamanlı uygulamalardır bu da onu kamera taşıyan araçlarda kullanmak için ideal kütüphanelerden biri yapar.[17]

3.3.5 Meshroom Nedir ?

Meshroom açık kaynaklı bir fotogrametrik bilgisayar görüşü frameworkü olan Alicevision'u kullanan açık kaynaklı bir fotogrametri boru hattı uygulamasıdır. Boru hattının her aşamasının özelleştirilmesine imkan sağlaması programın güçlü taraflarındandır.Meshroom aynı zamanda üretilen nokta bulutunun dokuya çevrilmesi ve çevrilen Programın orijinal versiyonu CUDA gerektirmektedir. Bu yüzden programın sadece Nvida marka ekran kartları içeren bilgisayarlarda çalışabilmektedir. Fakat açık kaynaklı olması sebebiyle topluluk bu soruna çözüm olarak MeshroomCL adlı versiyonu geliştirmiştir ve bu versiyon CUDA ile yapılan işlemlerin OpenCL'e portlandığı tüm bilgisayarlarda kullanılabilen bir sürümdür.

3.3.6 Neden Proje Sırasında Airsim Tercih Edildi ?

ROS, açık kaynaklı bir proje olarak 2007 yılından beri geliştirilen daha oturmuş bir simülasyon ortamı olmasına karşın yeni kullanıcılar için ilk etapta öğrenmesi Airsim'e göre daha zordur. Örneğin Airsim için bir simülasyonu drone veya araba modunda çalıştırmak bir onay penceresi üzerinden yapılabilir ve seçilen aracın konfigürasyonu tek bir json dosyasını düzenleyerek özelleştirilebilir. Bu şekildeki temel ayarlamaların kolaylıkla yapılabilmesi projeye harcanan zaman içerisinde simülasyon ortamının kullanımının öğrenilmesinin payını küçülterek asıl odak noktası olan 3 boyutlu modellemenin geliştirilmesine daha fazla zaman ayrılmasına imkan sağladı.

4. Yapılan Çalışmalar

Yapacağımız çalışmayı Airsim ortamında denemeyi kararlaştırdık. Airsim simülasyon ortamında bir Drone'u elimizle uçurduk, Drone'un üzerine bir Lidar sensörü yerleştirdik. Bunu yaparken setting.json dosyası üzerinden aşağıdaki kod parçası ile eklediğimiz Lidar sensörünün farklı özelliklerine erişim sağladık.

```
            "Sensors": {
                "LidarSensor1": {
                    "SensorType": 6,
                    "Enabled" : true,
                    "NumberOfChannels": 16,
                    "RotationsPerSecond": 10,
                    "PointsPerSecond": 10000,
                    "X": 0, "Y": 0, "Z": -1,
                    "Roll": 0, "Pitch": 0, "Yaw" : 0,
                    "VerticalFOVUpper": -15,
                    "VerticalFOVLower": -25,
                    "HorizontalFOVStart": -50,
                    "HorizontalFOVEnd": 50,
                    "DrawDebugPoints": true,
                    "DataFrame": "SensorLocalFrame"
                },
```

Şekil 3. Lidar Konfigürasyonu

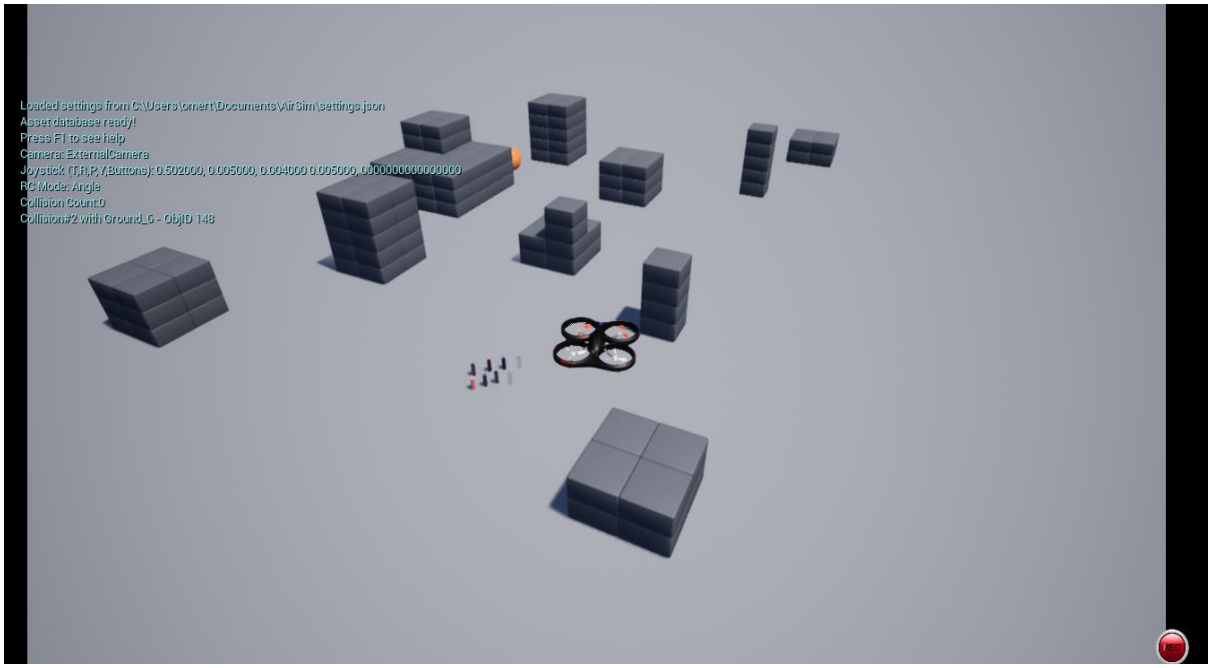
Daha sonra Lidar sensörü yerleştirdiğimiz Drone'umuzu haritada bulunan geometrik şekillerin üzerinden uçurduk. Yazdığımız kod sayesinde uçuş sırasında aynı anda Lidar verilerini topladık bunları bir dosyaya kaydettik.

```
def lidar_execute():
    print('Scanning Has Started\n')
    print('Use Keyboard Interrupt \'CTRL + C\' to Stop Scanning\n')
    existing_data_cleared = False   #change to true to superimpose new scans onto existing .asc files
    try:
        while record_data:
            file_name_with_ext = f"{filename}.asc"
            if not existing_data_cleared:
                f = open(file_name_with_ext,'w')
            else:
                f = open(file_name_with_ext,'a')
            lidar_data =lidar_client.getLidarData(vehicle_name='Drone1',lidar_name='LidarSensor2')

            orientation = lidar_data.pose.orientation
            q0, q1, q2, q3 = orientation.w_val, orientation.x_val, orientation.y_val, orientation.z_val
            rotation_matrix = np.array(([1-2*(q2*q2+q3*q3),2*(q1*q2-q3*q0),2*(q1*q3+q2*q0)],
                                        [2*(q1*q2+q3*q0),1-2*(q1*q1+q3*q3),2*(q2*q3-q1*q0)],
                                        [2*(q1*q3-q2*q0),2*(q2*q3+q1*q0),1-2*(q1*q1+q2*q2)]))

            position = lidar_data.pose.position
            for i in range(0, len(lidar_data.point_cloud), 3):
                xyz = lidar_data.point_cloud[i:i+3]

                corrected_x, corrected_y, corrected_z = np.matmul(rotation_matrix, np.asarray(xyz))
                final_x = corrected_x + position.x_val
                final_y = corrected_y + position.y_val
                final_z = corrected_z + position.z_val

                f.write("%f %f %f\n" % (final_x,final_y,final_z))
            f.close()
            #time.sleep(0.2)
            existing_data_cleared = True
    except KeyboardInterrupt:
        airsim.wait_key('Press any key to stop running this script')
        print("Done!\n")
    finally:
        return
```

Şekil 4.1. Lidar Sensörünün Uçuş Sırasında Veri Kaydetmesini Sağlayan Kod Parçası

```python
if __name__ == "__main__":
    filename = "kosegen_2"
    lidarThread = Thread(target=lidar_execute,daemon=True)

    client.confirmConnection()
    lidar_client.confirmConnection()
    client.enableApiControl(False)
    client.enableApiControl(True)
    client.armDisarm(False)

    airsim.wait_key('Kalkis icin bir tusa basiniz')
    client.armDisarm(True)
    client.takeoffAsync().join()
    airsim.wait_key('Veri toplama ve ilerleme icin bir tusa basiniz')
    client.moveToZAsync(-35, 5).join()
    record_data = True
    lidarThread.start()
    x_pos =  client.getMultirotorState().kinematics_estimated.position.x_val
    y_pos =  client.getMultirotorState().kinematics_estimated.position.y_val
    z_pos = client.getMultirotorState().kinematics_estimated.position.z_val

    client.moveToPositionAsync( x_pos+ 30, y_pos+30, z_pos, 2).join()
    client.moveToPositionAsync( x_pos, y_pos, z_pos, 2).join()
    client.hoverAsync().join()

    airsim.wait_key("inis yapmak icin bir tusa basiniz")
    client.moveToZAsync(-1, 5).join()
    client.landAsync().join()
    #client.armDisarm(False)
    record_data = False
    client.enableApiControl(False)

    pass
```

Şekil 4.2. Lidar Sensörünü Kullanan Ana Program

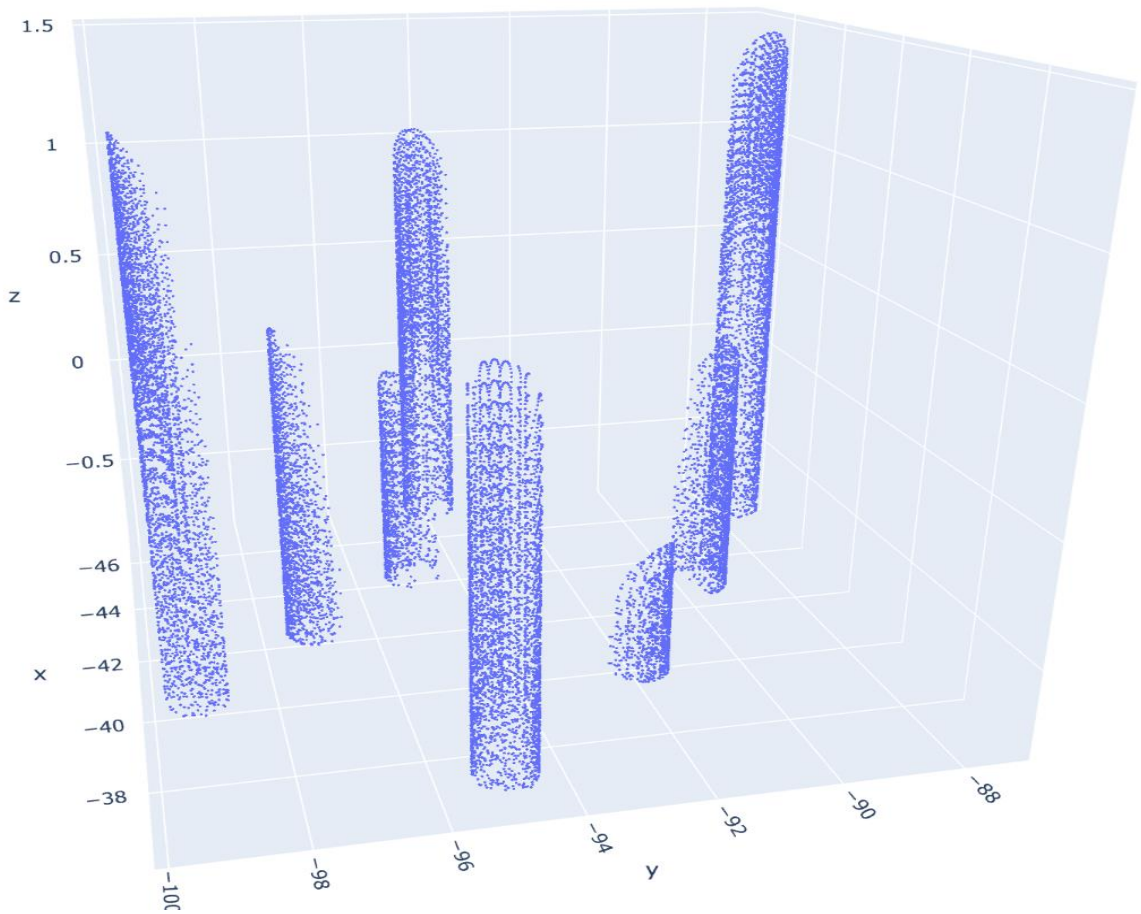


Şekil 5. Blocks Haritasından Bir Fotoğraf

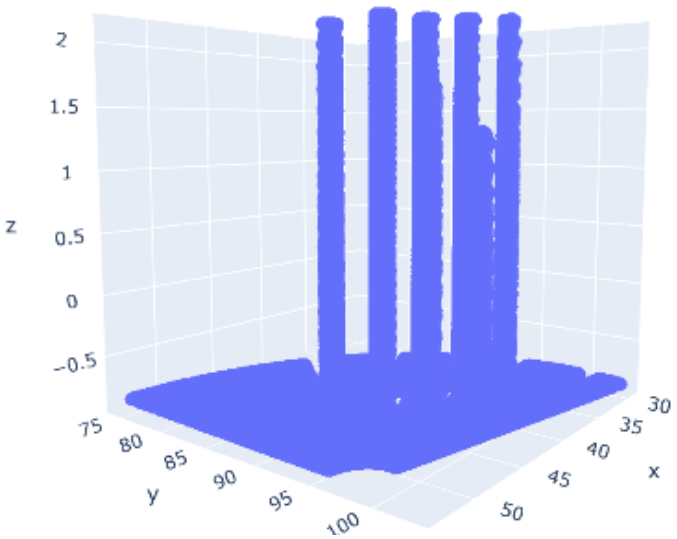
Topladığımız lidar verilerinden oluşturduğumuz nokta bulutunu PyVista ve pandas kütüphanelerini kullanarak görüntüledik. Ayrıca aynı kütüphane sayesinde nokta bulutlarına meshing işlemi uygulayarak doku elde etmeyi başardık.

```python
import pandas
import numpy as np
import pyvista as pv

column_names = ["x","y","z"]
points = pandas.read_csv("kosegen_2.asc",delimiter = " ",names=column_names)
points.drop(points[points.z > 0.8].index, inplace=True)

numpy_arr = points.to_numpy()
sample  = 10

cloud = pv.wrap(numpy_arr)
cloud.plot()

volume = cloud.reconstruct_surface()
volume.plot()

print("end of program")

```

Şekil 6. Lidar Verilerini Görselleştiren Kod Parçası

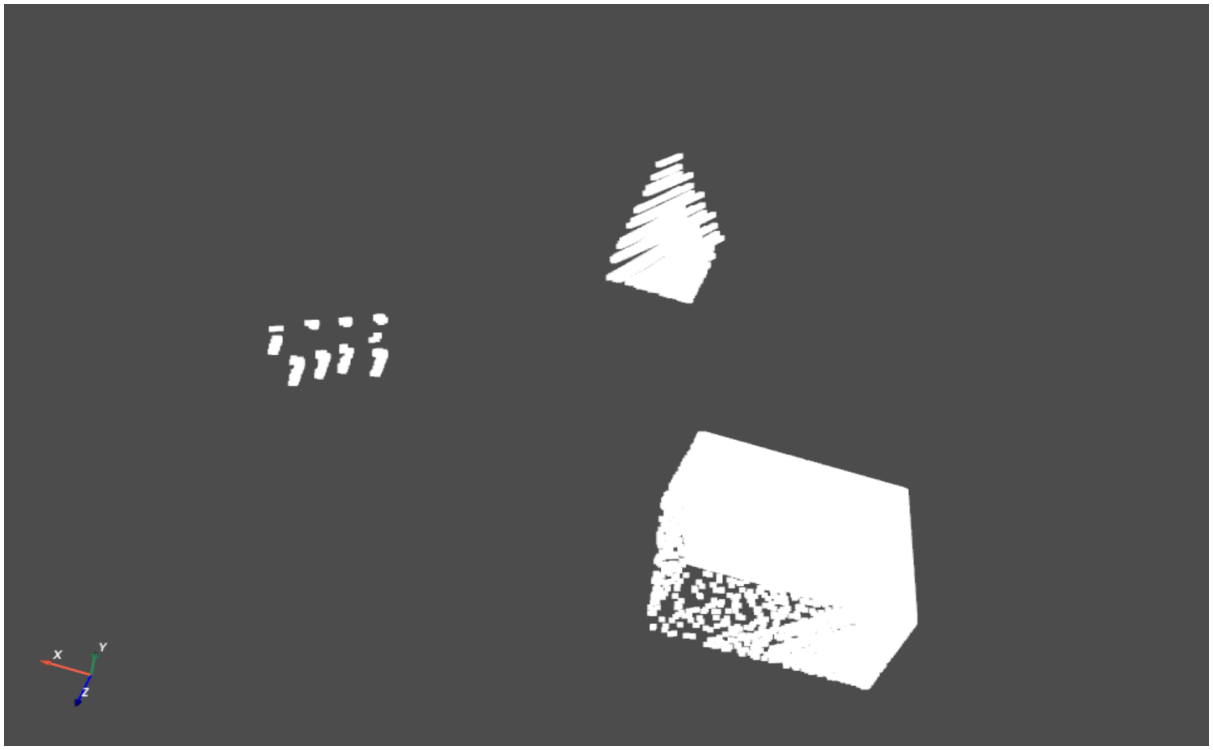


Şekil 7.1. Lidar Nokta Bulutu

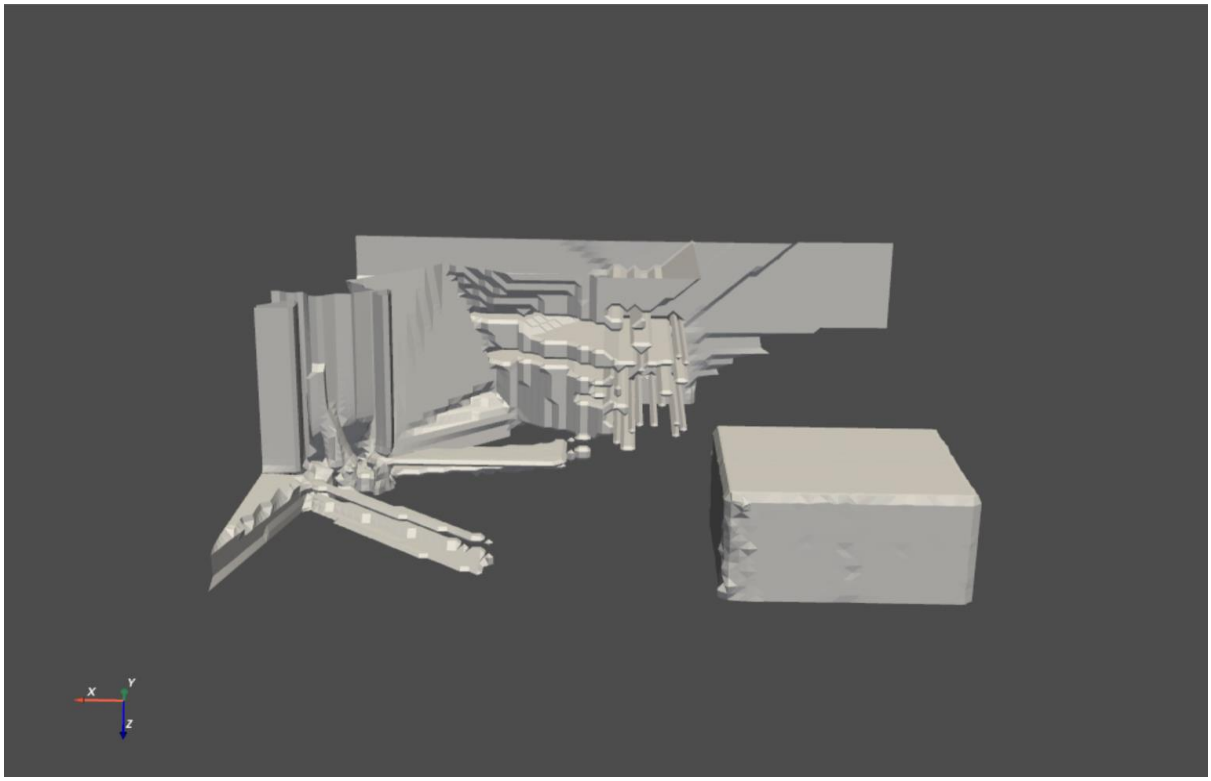


Şekil 7.2. Lidar Nokta Bulutu

Ancak PyVista kullanarak elde ettiğimiz sonuçlar istediğimiz seviyeye ulaşamadı bu yüzden farklı metodlar denemeye karar verdik.



Şekil 8.1. PyVista Üzerinde İşlenmemiş Nokta Bulutu



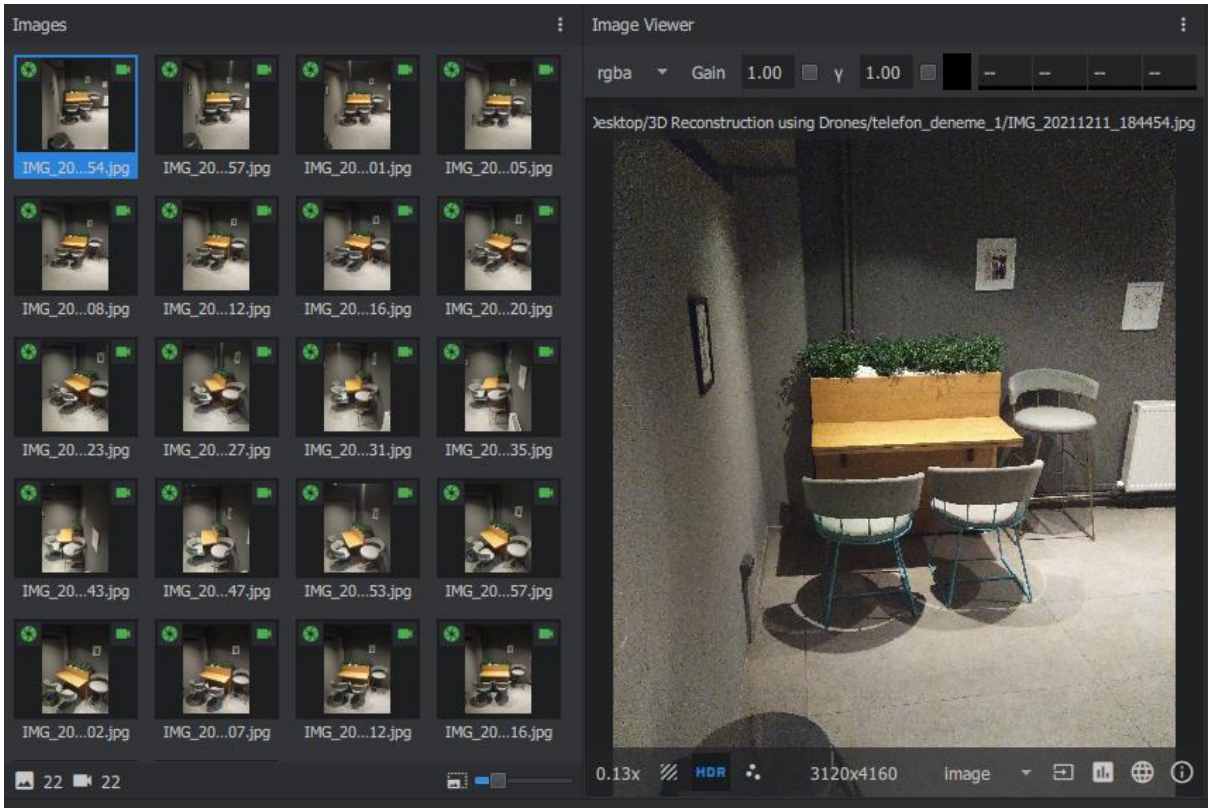
Şekil 8.2.Nokta Bulutunun Meshing İşlemi Yapılmış Hali

Lidar verilerini toplamak ve işlemek karmaşıklık açısından daha kolay olsa da renk ve doku bilgisi üretmemesi modelleme işlemleri için büyük dezavantajlar oluşturmaktadır. Bu sebeple projemizin bir sonraki aşamasında fotogrametri kullanarak 3b modellemeye geçmeye karar verdik. Bunun ilk şartı olarak da Drone'umuzun donanımına bir kamera ekledik.

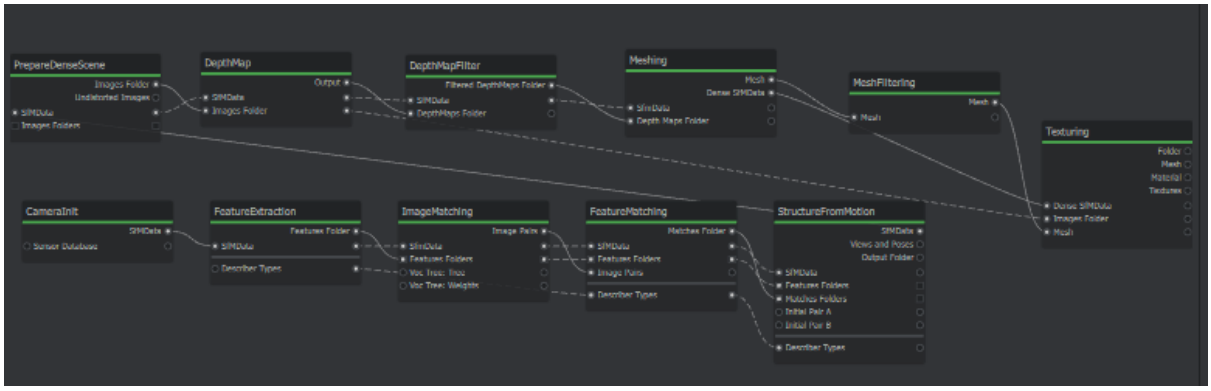
```
{
    "SeeDocsAt": "https://microsoft.github.io/AirSim/settings/",
    "SettingsVersion": 1.2,

    "SimMode": "Multirotor",

     "Vehicles": {
        "Drone1": {
            "VehicleType": "simpleflight",
            "AutoCreate": true,
            "Cameras" : {
              "high_res_bottom": {
                  "CaptureSettings" : [
                      {
                          "ImageType" : 0,
                          "Width" : 1920,
                          "Height" : 1080
                      }
                  ],
                  "X": 0, "Y": 0, "Z": 0,
                  "Roll": 0, "Pitch": -90, "Yaw" : 0
                }
            },
```

Şekil 9.Kamera Konfigürasyonu

Daha sonra Meshroom adlı program aracılığıyla elde ettiğimiz fotoğraf görüntülerini işlemeye karar verdik. Uygulamamızın çalıştığından emin olmak için gerçek dünyada bir kafeye gidip orada bir çalışma masasının farklı noktalarından çekilmiş 22 farklı fotoğrafı kullanarak bir deneme yapmaya karar verdik.

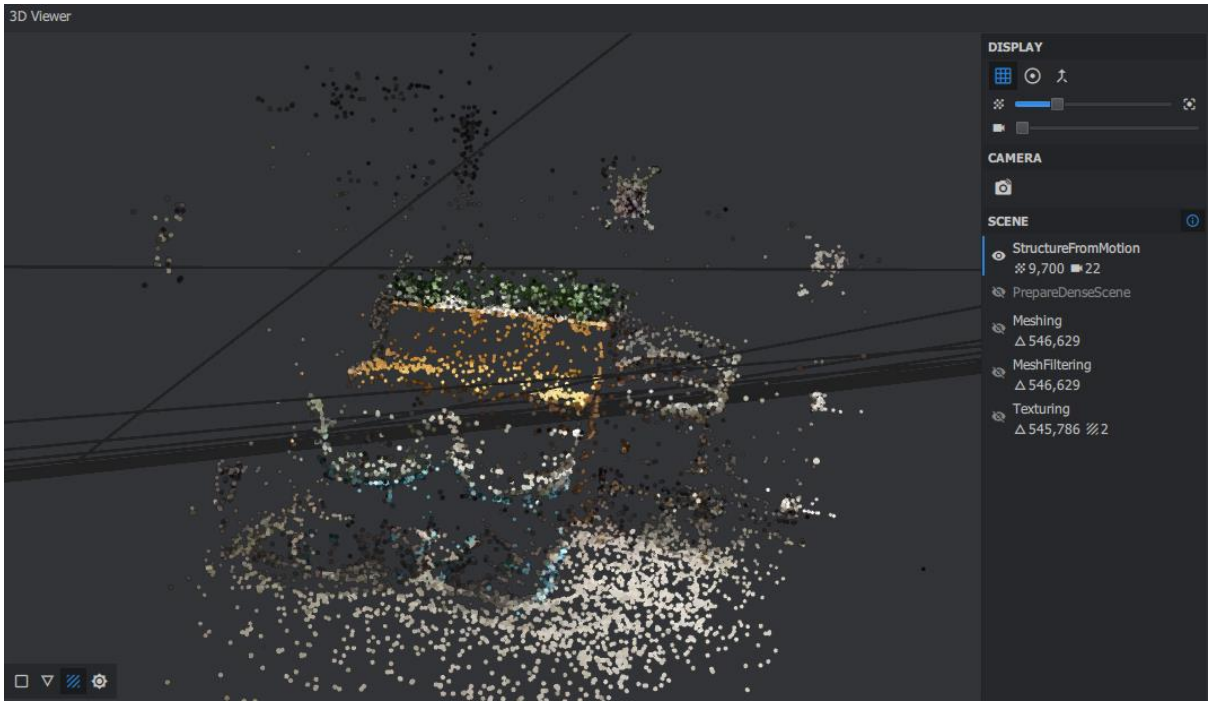


Şekil 10.Meshroom Üzerinde İşlenmemiş Kafe Fotoğrafları

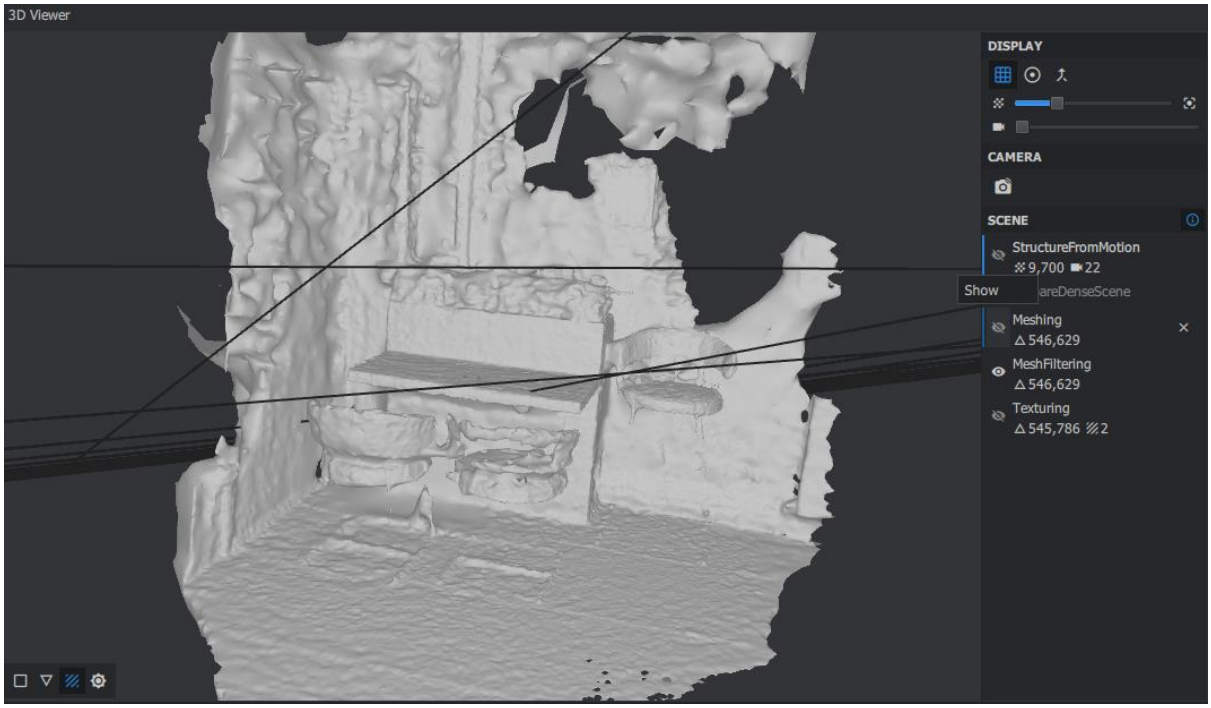


Şekil 11.Fotogrametri Boru Hattı

Çektiğimiz fotoğrafları meshroom içindeki projeye ekledik ve Structure From Motion boru hattını çalıştırdık. Art arda yapılan işlemler sonucunda uygulama bize nokta bulutu, mesh ve üzerine doku uygulanmış meshleri çıktı olarak verdi. Bu sayede fotogrametri çalışmalarımızda meshroom programını kullanabileceğimizi teyit etmiş olduk.



Şekil 11.Kafe Masasının Nokta Bulutu Hali



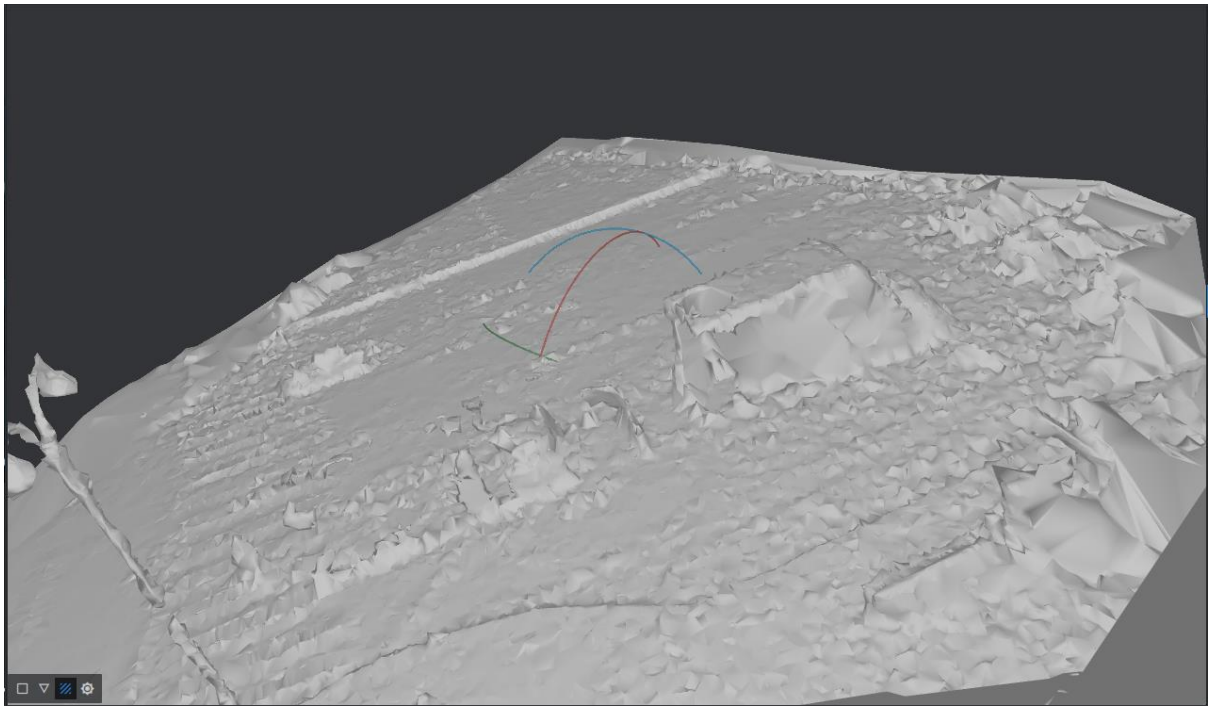
Şekil 12.Kafe Masasının Nokta Bulutundan Üretilen Mesh

Şekil 13.Kafe Masasının 3B Modellenmiş Hali

Simülasyon ortamında fotogrametri çalışması yapmak için ilk girişimimizi Lidar denemelerimizi yaptığımız geometrik cisimlerden oluşan blocks ortamında denedik. Fakat blocks dünyasında cisimlerin üzerinde dokular bulunmadığı için lidar ile yapabildiğimiz modellemeyi fotogrametri yöntemleriyle yapmamız mümkün olmadı. Bu sebeple doku içeren cisimlerden oluşan şehir dünyasına geçiş yaptık.

Şekil 14.CityEngine Şehir Haritasında Çekilmiş Bir Fotoğraf

Burada ilk denememizi kameranın Dronun tam altına baktığı açıyla bir otoparkın üzerinden uçarak yaptık ama sonuçlar beklediğimiz gibi olmadı.



Şekil 15.Otopark Uçuşundan Elde Edilen Mesh

Dolayısıyla modelimize derinlik algısı katabilmek için hedefimizin etrafında bir yörünge çizmemiz gerektiğini fark ettik. Fakat kamera Dronun tam altına bakarken yörüngede hareket etmemiz derinlik algısına katkıda bulunmayacaktır. Bu sebeple kamera bakış açısının Dronun aşağısından ön-aşağı bakan bir bakış açısında duracak şekilde değiştirdik. Bu sayede yörünge merkezine aldığımız bölgenin önceden sadece üstten çekebildiğimiz fotoğraflarını çapraz bir açıdan üst ve yan yüzleri içerecek şekilde çekebilmeye başladık. Bu sayede çekilen fotoğrafların derinlik algısı istediğimiz ölçüye geldi.

```
"SimMode": "Multirotor",

"Vehicles": {
   "Drone1": {
       "VehicleType": "simpleflight",
       "AutoCreate": true,
       "Cameras" : {
         "high_res_bottom": {
             "CaptureSettings" : [
                 {
                     "ImageType" : 0,
                     "Width" : 2048,
                     "Height" : 1080
                 }
             ],
             "X": 0, "Y": 0, "Z": 0,
             "Roll": 0, "Pitch": -45, "Yaw" : 0
```

Şekil 16.Güncel Kamera Parametreleri

Daha sonra Dronun yörünge çizmesini sağlayacak gerekli kodu yazdık.

```
168     def start_orbit(self):
169         self.libMode = True
170         self.do_initial_calculations()
171         print("arming the drone...")
172         self.client.armDisarm(True)
173 
174         # AirSim uses NED coordinates so negative axis is up.
175         start = self.client.getMultirotorState().kinematics_estimated.position
176         z = start.z_val # use current altitude
177 
178         #print("climbing to position: {},{},{}".format(start.x_val, start.y_val, z))
179         self.client.moveToPositionAsync(start.x_val, start.y_val, z, self.speed).join()
180         self.z = z
181 
182         #print("ramping up to speed...")
183         count = 0
184         self.start_angle = None
185         self.next_snapshot = None
186 
187         # ramp up time
188         ramptime = self.radius / 10
189         self.start_time = time.time()
190 
191         while count < self.iterations:
192             if self.snapshots > 0 and not (self.snapshot_index < self.snapshots):
193                 break
194             # ramp up to full speed in smooth increments so we don't start too aggressively.
195             now = time.time()
196             speed = self.speed
197             diff = now - self.start_time
198             if diff < ramptime:
199                 speed = self.speed * diff / ramptime
200             elif ramptime > 0:
201                 print("reached full speed...")
202                 ramptime = 0
203 
204             lookahead_angle = speed / self.radius
```

Şekil 17.1.Orbit Hareketini Sağlayan Kod Parçası

```
205 
206             # compute current angle
207             pos = self.client.getMultirotorState().kinematics_estimated.position
208             dx = pos.x_val - self.center.x_val
209             dy = pos.y_val - self.center.y_val
210             actual_radius = math.sqrt((dx*dx) + (dy*dy))
211             angle_to_center = math.atan2(dy, dx)
212 
213             camera_heading = (angle_to_center - math.pi) * 180 / math.pi
214 
215             # compute lookahead
216             lookahead_x = self.center.x_val + self.radius * math.cos(angle_to_center + lookahead_angle)
217             lookahead_y = self.center.y_val + self.radius * math.sin(angle_to_center + lookahead_angle)
218 
219             vx = lookahead_x - pos.x_val
220             vy = lookahead_y - pos.y_val
221 
222             if self.track_orbits(angle_to_center * 180 / math.pi):
223                 count += 1
224                 print("completed {} orbits".format(count))
225 
226             self.camera_heading = camera_heading
227             self.client.moveByVelocityZAsync(vx, vy, z, 1, airsim.DrivetrainType.MaxDegreeOfFreedom, airsim.YawMode(False, camera_heading)
228 
229         self.client.moveToPositionAsync(start.x_val, start.y_val, z, 2).join()
```

Şekil 17.2.Orbit Hareketini Sağlayan Kod Parçasının Devamı

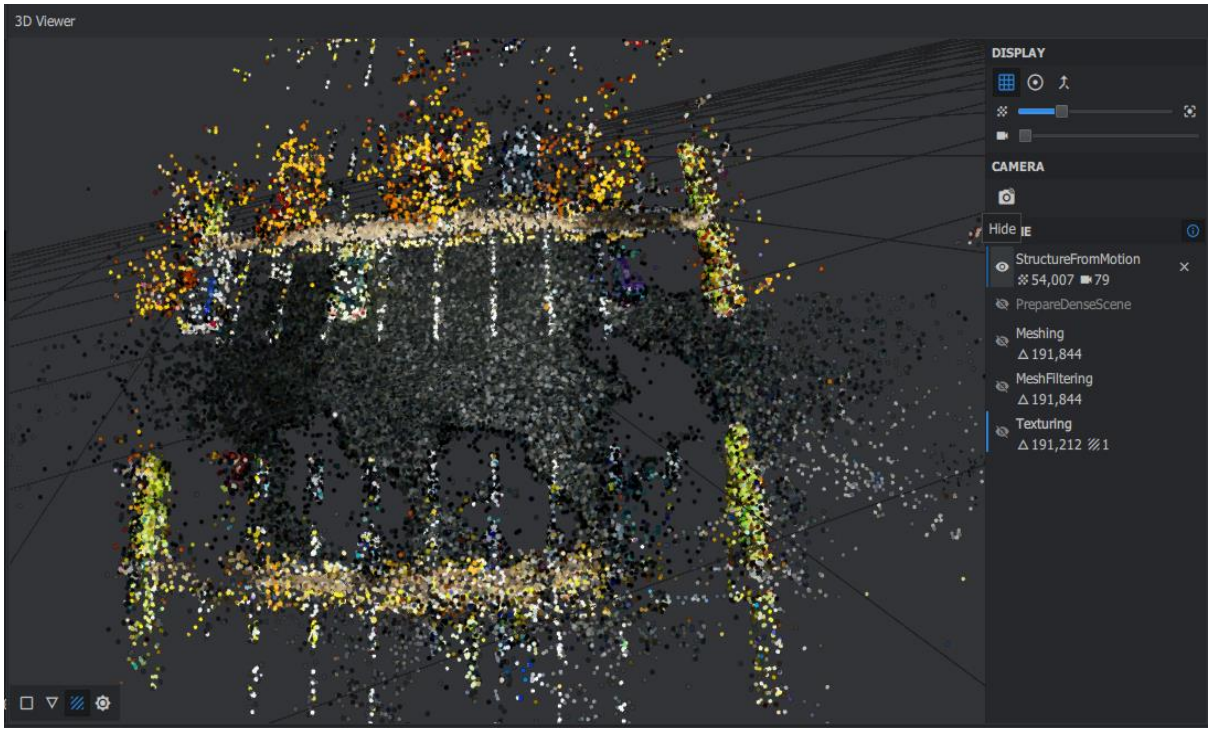
```python
if __name__ == "__main__":
    filename = "orbit_test_1"

    cameraThread = Thread(target=camera_execute,daemon=True)

    client.confirmConnection()
    camera_client.confirmConnection()
    client.enableApiControl(False)
    client.enableApiControl(True)
    client.armDisarm(False)

    airsim.wait_key('Kalkis icin bir tusa basiniz')
    client.armDisarm(True)
    client.takeoffAsync().join()

    airsim.wait_key('Yukselis icin bir tusa basiniz')
    client.moveToZAsync(-5, 5).join()
    client.hoverAsync().join()

    airsim.wait_key('Veri toplama ve yorunge icin bir tusa basiniz')
    record_data = True
    drone_posistion = client.getMultirotorState().kinematics_estimated.position
    current_altitude = drone_posistion.z_val
    orientation = client.getMultirotorState().kinematics_estimated.orientation

    #ACI HESAPLARI BASLANGIC :
    #convert quaternion to euler
    roll_x, pitch_y, yaw_z = euler_from_quaternion(orientation.x_val, orientation.y_val, orientation.z_val, orientation.w_val)

    yaw_z = yaw_to_degrees(yaw_z)
    print(roll_x, pitch_y, yaw_z)
    point_a = orientation.x_val + 10*math.cos(yaw_z*math.pi/180)
    point_b = orientation.y_val + 10*math.sin(yaw_z*math.pi/180)
    print(point_a, point_b)
    center_vector = [point_a-orientation.x_val,point_b-orientation.y_val]
    print(center_vector)
```

Şekil 17.3.Orbit Hareketini Sağlayan Kod Parçasının Devamı

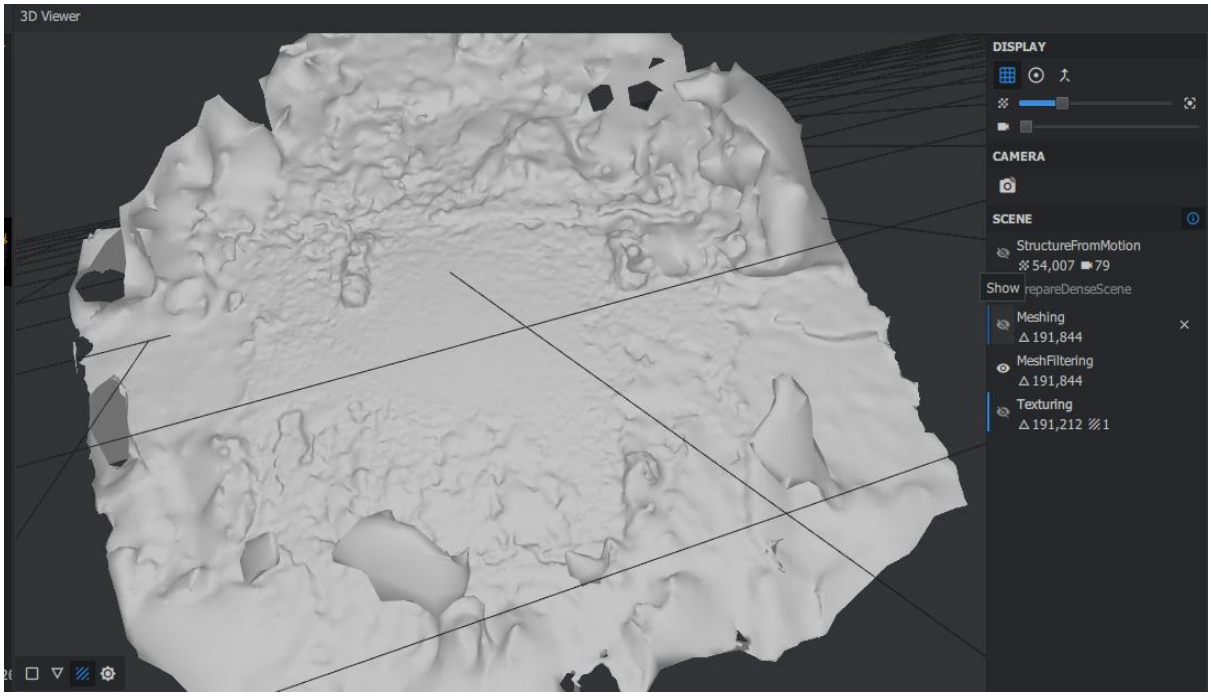
```python
    #ACI HESAPLARI BITIS
    w_val = orientation.w_val
    orbit_radius = 10
    nav = orbit.OrbitNavigator(client =client,radius=orbit_radius,altitude=current_altitude, speed=2, iterations=1, center = center_vector, snapshots=0)
    nav.start_orbit()

    airsim.wait_key("inis yapmak icin bir tusa basiniz")
    record_data = False
    client.moveToZAsync(-1, 5).join()
    client.landAsync().join()
    client.enableApiControl(False)

    pass
```

Şekil 17.4.Orbit Hareketini Sağlayan Kod Parçasının Devamı

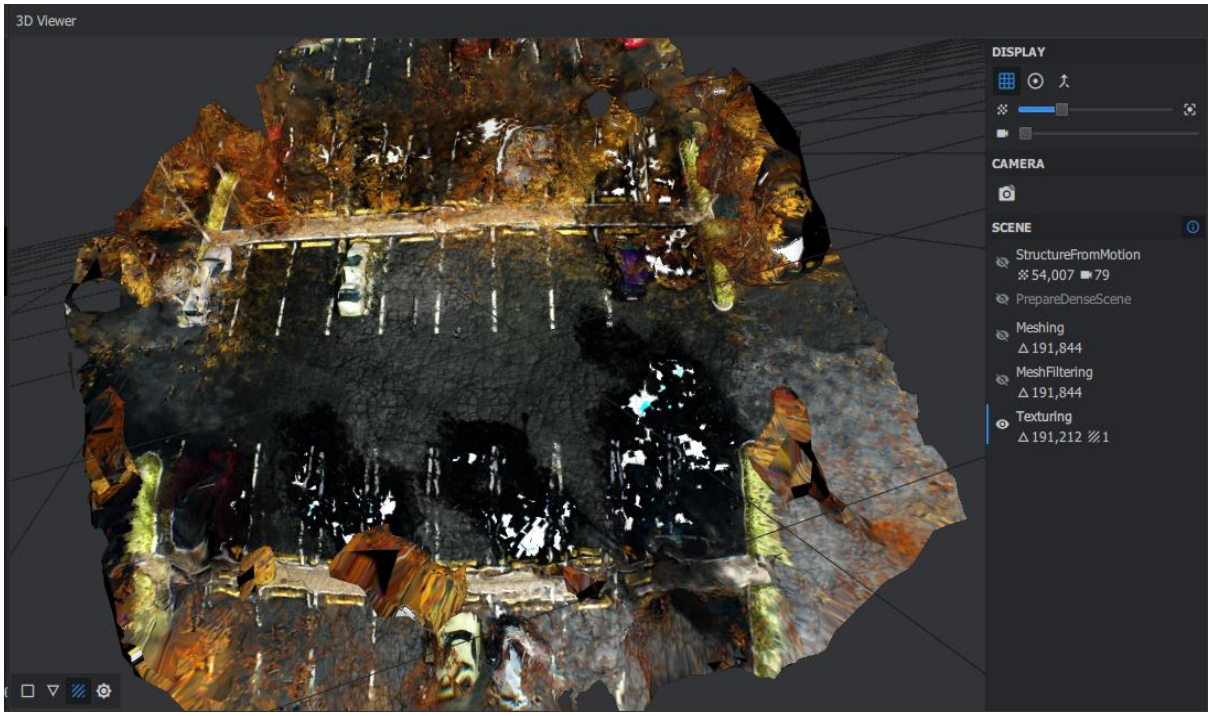
Aynı otopark üzerinde yaptığımız uçuştan topladığımız fotoğraflar sayesinde 3b derinlik algısı taşıyan bir model elde ettik.



Şekil 18.1.Otopark Üzerinden Elde Edilmiş Nokta Bulutu



Şekil 18.2.Otopark Nokta Bulutu Üzerinden Elde Edilmiş Mesh



Şekil 18.3.Otopark Nokta Bulutu Üzerinden Elde Edilmiş 3B Model

5. Sonuç

Yaptığımız çalışmaların sonucunda 3D rekonstrüksiyon için ilk başta belirli bir bölge üzerinde tepeden fotoğraf çekiminin, otopark gibi benzer şekillerin tekrarından oluşan bölgelerde programımızın istenilen performansı veremediği görüldü. Buna çözüm olarak sabit bir noktaya dönük şekilde yörünge oluşturuldu ve noktanın bu yörüngeden fotoğrafları çekildi. Bu sayede cisimlerin farklı yüzlerinin fotoğrafları çekilerek derinlik algısı ve cisimlerin 3D rekonstrüksiyonu geliştirildi. Biraz daha düzgün sonuçlar elde edildi. İlerleyen süreçte fotoğraflardan elde ettiğimiz modelleri geliştirmek için farklı hareket algoritmaları ve kamera ayarları tekrar denenecek. Ardından ise Lidar kullanarak aldığımız veriler ile fotogrametri tekniği kullanarak elde ettiğimiz veriler birleştirilip ortak ve daha gelişmiş bir model elde etmek üzerine çalışmalar yürüteceğiz.

Kaynakça

[1]Moons, Theo, Luc Van Gool, and Maarten Vergauwen. "3D reconstruction from multiple images part 1: Principles." Foundations and Trends in Computer Graphics and Vision 4.4 (2010): 287-404.

[2]Lee, Jisang & Hong, Seunghwan & Cho, Hanjin & Park, Ilsuk & Cho, Hyoungsig & Sohn, Hong-Gyoo. (2015). Accuracy Comparison Between Image-based 3D Reconstruction Technique and Terrestrial LiDAR for As-built BIM of Outdoor Structures. Journal of the Korean Society of Surveying, Geodesy, Photogrammetry and Cartography. 33. 557-567. 10.7848/ksgpc.2015.33.6.557.

[3]Fawad Ahmad, Christina Shin, Eugene Chai, Karthik Sundaresan, and Ramesh Govindan. ARES: Accurate, Autonomous, Near Real-time 3D Reconstruction using Drones arXiv: 2104.08634, 2021.

[4]S. Nair, A. Ramachandran and P. Kundzicz, "Annotated reconstruction of 3D spaces using drones," 2017 IEEE MIT Undergraduate Research Technology Conference (URTC), 2017, pp. 1-5, doi: 10.1109/URTC.2017.8284202.

[5]N. YASTIKLI Et Al. , "Fotogrametrik Harita ve LiDAR Verileri ile 3B Kent Modeli Üretimi," 16. Türkiye Harita Bilimsel ve Teknik Kurultayı , Ankara, Turkey, pp.1, 2017

[6]N. Polat , "LiDAR verilerinin CSF algoritmasıyla filtrelenmesi ve Sayısal Arazi Modeli üretimi", Türkiye Lidar Dergisi, c. 1, sayı. 1, ss. 21-25, Ara. 2019

[7]H. İ. Şenol , A. Y. Yiğit , Y. Kaya ve A. Ulvi , "İHA ve yersel fotogrametrik veri füzyonu ile kültürel mirasın 3 boyutlu (3B) modelleme uygulaması: Kanlıdivane Örneği", Türkiye Fotogrametri Dergisi, c. 3, sayı. 1, ss. 29-36, Haz. 2021, doi:10.53030/tufod.935920

[8]A. Ulvi , M. Yakar , A. Y. Yiğit ve Y. Kaya , "İha Ve Yersel Fotogrametrik Teknikler Kullanarak Aksaray Kızıl Kilisenin 3b Modelinin Ve Nokta Bulutunun Elde Edilmesi", Geomatik, c. 5, sayı. 1, ss. 19-26, Nis. 2020, doi:10.29128/geomatik.560179

[9]Tercan, E. (2017). İNSANSIZ HAVA ARACI KULLANILARAK ANTİK KENT VE TARİHİ KERVAN YOLUNUN FOTOGRAMETRİK BELGELENMESİ: SARIHACILAR ÖRNEĞİ . Mühendislik Bilimleri ve Tasarım Dergisi , 5 (3) , 633-642 . DOI: 10.21923/jesd.315232

[10]https://wingtra.com/drone-photogrammetry-vs-lidar/

[11]https://evrimagaci.org/lidar-teknolojisi-nedir-nasil-calisir-nerelerde-kullanilir-10214

[12]O. Muratov, Y. Slynko, V. Chernov, M. Lyubimtseva, A. Shamsuarov and V. Bucha, "3DCapture: 3D Reconstruction for a Smartphone," 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2016, pp. 893-900, doi: 10.1109/CVPRW.2016.116.

[13]http://wiki.ros.org/ROS/Introduction

[14]http://gazebosim.org/

[15]https://microsoft.github.io/AirSim/

[16]Pulli, Kari & Baksheev, Anatoly & Kornyakov, Kirill & Eruhimov, Victor. (2012). Real-Time Computer Vision with OpenCV. Communications of the ACM. 55. 61-69. 10.1145/2184319.2184337.

[17]https://docs.opencv.org/4.5.3/